2019/4/22 - 2019/4/28 Weekly Report

2019年4月22日 10:47

回顾

1. 概况

- 创建联邦学习资料库的Github Repo https://github.com/ZeroWangZY/federated-learning/blob/master/README.md;
- 学习联邦学习的具体算法

2. 时间安排

时间	主要内容	实验室工作时长
周一	实验室	8h
周二	实验室	8h
周三	实验四	8h
周四	实验室、返校	4h
周五	在校上课	0
周六	在校	0
周日	返回实验室	4h

3. 联邦学习

1. FederatedAveraging Algorithm算法

来源

> Communication-Efficient Learning of Deep Networks from Decentralized Dat

该算法是google最早提出联邦算法概念时给出的，现在用于很多

简介

在典型的机器学习系统中，诸如随机梯度下降（SGD）的优化算法是云服务器上均匀分配的大型数据集上运行。这种高度迭代的算法在连接训练数据时，需要低延迟和高吞吐量。但在“联合学习”的情况下，数据以非常不均匀的方式分布在成百上千万台的设备上。此外，这些设备明显具有更高的延迟、低吞吐量连接，并且只能间歇性地训练。

联合平均算法，与朴素联合 SGD 版本相比，联合平均算法训练深度网络的通信需求要少 10-100倍。关键的思路是，在现代移动设备中使用强大的处理器来计算比简单梯度步骤更高质量的更新。由于生成好模型需要的高质量更新的迭代次数少了，训练需要的通信也就少了。考虑到上传速度通常比下载速度慢得多，因此我们还开发出一种新的方法，通过使用随机旋转（rotation）和量化（quantization）来压缩更新，将上传通信成本降低了 100 倍。

前置知识

SGD：https://blog.csdn.net/a130737/article/details/42343699

$$\theta^{(t)} \leftarrow \theta^{(t-1)} - \epsilon_t \frac{1}{B} \sum_{t'=Bt+1}^{B(t+1)} \frac{\partial L(z_{t'}, \theta)}{\partial \theta}.$$

B = the training set size: standard gradient descent

B = some value between 1 and the training se size: minibatch gradient descent, a kind of SGD

B = 1: online gradient descent, SGD

The FederatedAveraging Algorithm

每个样本根据loss function计算的值

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w). \quad (1)$$

loss

最Naïve GD: loss function最小, w为参数

For a machine learning problem, we typically take $f_i(w) = \ell(x_i, y_i; w)$, that is, the loss of the prediction on example (x_i, y_i) made with model parameters w. We assume there are K clients over which the data is partitioned, with $\mathcal{P}_k$ the set of indexes of data points on client k, with $n_k = |\mathcal{P}_k|$. Thus, we can re-write the objective (1) as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w).$$

SGD：随机从样本中取nk个

If the partition $\mathcal{P}_k$ was formed by distributing the training examples over the clients uniformly at random, then we would have $\mathbb{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$, where the expectation is over the set of examples assigned to a fixed client k. This is the IID assumption typically made by distributed optimization algorithms; we refer to the case where this does not hold (that is, F_k could be an arbitrarily bad approximation to f) as the non-IID setting.

In the federated setting, there is little cost in wall-clock time to involving more clients, and so for our baseline we use large-batch synchronous SGD; experiments by Chen et al. [8] show this approach is state-of-the-art in the data center setting, where it outperforms asynchronous approaches. To apply this approach in the federated setting, we select a C-fraction of clients on each round, and compute the gradient of the loss over all the data held by these clients. Thus, C controls the *global* batch size, with $C = 1$ corresponding to full-batch (non-stochastic) gradient descent.[2] We refer to this baseline algorithm as `FederatedSGD` (or `FedSGD`).

C fraction，C=1即FedSGD

A typical implementation of FedSGD with $C = 1$ and a fixed learning rate η has each client k compute $g_k = \nabla F_k(w_t)$, the average gradient on its local data at the current model w_t, and the central server aggregates these gradients and applies the update $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$, since $\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$. An equivalent update is given by $\forall k,\ w_{t+1}^k \leftarrow w_t - \eta g_k$ and then $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$. That is, each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. Once the algorithm is written this way, we can add more computation to each client by iterating the local update $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$ multiple times before the averaging step. We term this approach FederatedAveraging (or FedAvg). The amount of computation is controlled by three key parameters: C, the fraction of clients that perform computation on each round; E, then number of training passes each client makes over its local dataset on each round; and B, the local minibatch size used for the client updates. We write $B = \infty$ to indicate that the full local dataset is treated as a single minibatch. Thus, at one endpoint of this algorithm family, we can take $B = \infty$ and $E = 1$ which corresponds exactly to FedSGD. For a client with n_k local examples, the number of local updates per round is given by $u_k = E\frac{n_k}{B}$; Complete pseudo-code is given in Algorithm 1.

平均gradient

第k个设备的数据产生的gk，生成中间参数，中间参数的平均值即新的w

FedAvg即增加此步，多一些循环次数

C：每一轮循环参与训练的比例

E：每个设备每轮循环中训练的次数权值，训练次数为

B：本地的batch size，B为无穷表示使用所有数据

当B为无穷且E为1时即FedSGD

Algorithm 1 FederatedAveraging. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

2. 纵向联邦学习的一种算法

见联邦学习笔记

计划

1. 下周

- 继续学习联邦学习，细节到具体算法；了解相关金融背景；为之后微众银行项目做准备；

2. 近期

- 系统学习可视化领域知识。目标：理解当前可视化领域中热门的研究方向。
- 学习GNN领域内容。目标：能够从原理出发理解GNN的应用领域及优势；能够在将算法应用在某些数据上。
- 熟悉联邦学习及相关金融领域知识，参与微众银行项目。